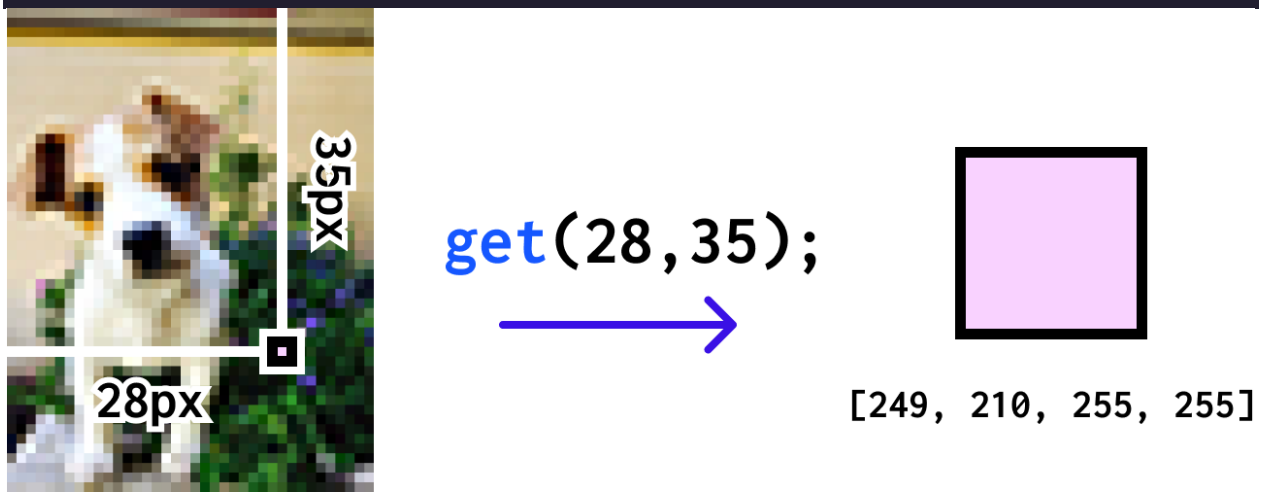
Introduction to Pixel Manipulation

16 min

We can also use p5.js to manipulate images and videos, down to the pixel. p5.js offers built-in functions to do this: `get()` and `set()`.

The `get()` function accesses the color of a specific pixel on the canvas. When given a pixel location, it returns the color of that pixel as an array of four numbers, representing the red, green, blue, and alpha (RGBA) values.

```
let pixelColor = get(28, 35); // Returns [r, g, b, a], where r, g, b, a are values between 0 and 255
```



Alternatively, the `get()` function also can retrieve regions of the canvas, returning them as a p5.js image element.

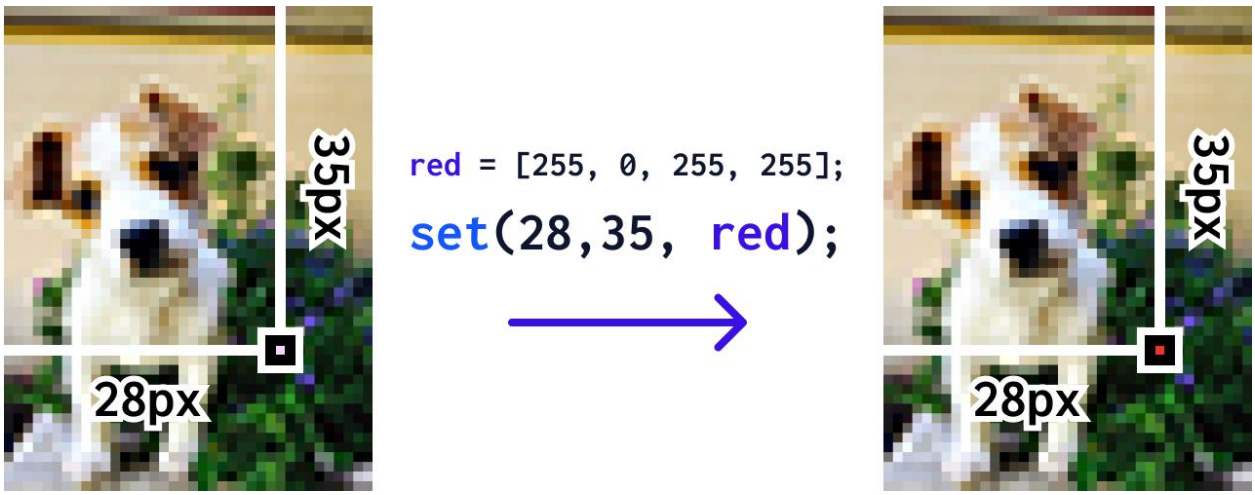
```
// Using get() to access a portion of canvas
let x = 0; let y = 0;
let w = 100; let h = 100;
let partOfCanvas = get(x, y, w, h); // Returns a region starting at (x, y) that's w-pixels wide and h-pixels high
```

When provided no arguments, it retrieves the entire canvas as a p5.js image element.

```
// Using get() to access entire canvas
let entireCanvas = get();
```

The `set()` function sets a pixel at a given location on the canvas to a new color.

```
let red = [255, 0, 0, 255];
set(28, 35, red); // Sets pixel at (28, 35) to red
```



The color can be in several formats. Above, we're using an array of 4 RGBA values, but you can also use a single greyscale value or a p5.js color element.

Just calling the `set()` function doesn't change what you see—to reflect changes onto the canvas, you need to use the `updatePixels()` function after the `set()` call.

```
let red = [255, 0, 0, 255];
set(0, 0, red); // Pixel is not actually changed on screen
updatePixels(); // Now pixel is changed!
```

If you use `set()` multiple times, calling `updatePixels()` once will reflect all those changes.

```
let red = [255, 0, 0, 255];
set(0, 0, red);
set(0, 1, red);
updatePixels(); // Both pixels are updated
```

The `get()` and `set()` functions also work as methods on individual image or video elements.

When using the `.set()` method on images or videos, you'll need to call it, along with the `.updatePixels()` method, before you draw them to the canvas.

```
let red = [255, 0, 0, 255];
img.set(50, 50, red); // Pixel is not actually changed in our image
element
img.updatePixels(); // Now pixel is changed in our pixel element
image(img, 0, 0); // Now our modified image element is on the screen
```

Because they're time-intensive, `get()` and `set()` are best suited for conveniently accessing a few pixels at a time. Later, we'll see how to do larger-scale

manipulations, such as accessing pixels in very large images and/or doing manipulations repeatedly in the `draw()` loop.

Instructions

1.

The p5.js Sketch to the right draws a smiley image that follows the movement of your mouse cursor.

In this exercise, we will use the `.get()` and `.set()` methods on the smiley's image element to change the black outlines in the image to blue ones. We will do this in `setup()`, so that we can draw the modified image later within the `draw()` loop.

First, in the `setup()` function, look for the double `for` loop that iterates through the pixels in the image. Inside the double `for` loop, use the `.get()` method to get the current pixel color, taking in the `i`, `j` position. Store the result as a new variable.
Hint
Remember that for individual image elements, `.get()` must be called as a method on the element itself.

```
img.get(x, y);
```

Because you're in a double `for` loop iterating through all the pixels, the current pixel location can be represented with (`i`,`j`).

2.

Knowing that the `get()` function returns a length-4 array of RGBA values, create an `if` statement that checks if the current pixel color is black.

Use the `isPixelBlack()` function defined at the bottom of **sketch.js**, which takes in a RGBA color array.
Hint
An `if` statement can be written as follows:

```
if (someCondition){
  //Do something here
}
```

The condition you're checking for is if `isPixelBlack()` returns `true`, when given the current pixel color.

3.

Using the `if` statement you just created, create logic that will set the color at the current pixel location to blue if it was originally black.

When setting the color, use the RGBA color array for the color blue: `[0, 0, 255, 255]`.

Hint

Remember the syntax for setting the pixel color for an image element:

```
img.set(x, y, c);
```

`c` is a color in the form of a length-4 RGBA color array, a p5.js color element, or a single grayscale value. In this example, use the color array representation of the color blue:

```
blue = [0, 0, 255, 255];
```

4.

Call the `.updatePixels()` method on the image after all pixels are iterated through, but before the image is drawn with the `image()` function.

Hint

Remember the syntax for updating pixels for an image element:

```
img.updatePixels();
```

You should call this after the double `for` loop is closed so it updates all changes made while looping, and call it before `image()` is called so that you draw the most updated version of the image.

sketch.js

```
let img;
let imagePath = 'https://static-assets.codecademy.com/Courses/Learn-p5/media/smiley.png';

function preload(){
  img = loadImage(imagePath);
}
function setup() {
  createCanvas(windowWidth, windowHeight);

  //Iterates through all pixels in the image
  for (let i = 0; i < img.width; i++){
    for (let j = 0; j < img.height; j++){
      //TODO: Get the color at the current pixel
```

```
      let pixel = img.get(i, j);
      //TODO: Check if the current pixel is black.
      //      If so, set it to blue.
      if (isPixelBlack(pixel)) {
        img.set(i, j, [0, 0, 255, 255]);
      }
    }
  }
  //TODO: Remember to update the pixels!
  img.updatePixels();
}

// isPixelBlack() takes in a  length-4 rgba color array,
// and returns true when the color is pure black,
// i.e. [0, 0, 0, 255]
// Examples:
//   isPixelBlack([0, 0, 0, 255]) == true
//   isPixelBlack([255, 0, 0, 255]) == false
function isPixelBlack(colorArray){
  return (colorArray[0] == 0 &&
     colorArray[1] == 0 &&
     colorArray[2] == 0 &&
     colorArray[3] == 255);
}

//Draw the smiley wherever you move your cursor
function draw() {
  image(img, mouseX, mouseY);
};

//Resize the canvas to the size of the window
function windowResized() {
  resizeCanvas(windowWidth, windowHeight);
}
```